Hearables: Deep Matched Filter for Online R-Peak Detection from In-Ear ECG in Mobile Application

Marek Żyliński[1], Harry J Davies[1], Qiyu Rao[1], and Danilo P Mandic[1]

[1] Department of Electrical and Electronic Engineering, Imperial College London, United Kingdom

Abstract

In-ear wearables, called Hearables, have been introduced for the monitoring of several physiological signals, inter alia ECG. However, in-ear ECG has a smaller amplitude and lower signal-to-noise ratio than standard ECG, which makes it difficult to automatically detect R peaks with standard algorithms. The solution for these problems can be the use of specially designed R-peak detectors, such as a Deep matched filter (DMF). The DMF is a deep neural network model designed to search for matches with an ECG template pattern in the input signal. The DMF operates as a neural network Matched Filter, trained to analyse in-ear data. In this study, we deploy the DMF in mobile application for online R-peak detection from in-ear ECG. We transfer the DMF into the Android mobile application. We conduct a computation time estimation for the implementation and analysis of DMF vulnerability to various types and levels of artefacts that can disturb ECG signal. We found that the baseline wander artifacts slightly impact the model's performance. Motion electrode and muscle artifices disturbed the model's sensitivity and positive predictive value (PPV) when signal to noise ratio drop bellow 6 dB. The average time of the single feed-forward model run, analysing 2 seconds of an ECG, on Samsung Galaxy Tab S8 was 7.46 ms. The model can be used for online detection of R-peaks in ear-ECG.

1. Introduction

In-ear wearable devices called Hearables, are small, comfortable, and discrete. They have been introduced for the monitoring of physiological signals including the electroencephalogram (EEG) [1], electrocardiogram (ECG) [2], and photoplethysmogram (PPG) [3]. However, in-ear ECG has a smaller amplitude than standard ECG [4], and a lower signal-to-noise ratio which makes it difficult to automatically detect R-peaks with standard algorithms [5]. Performance of automatic heartbeat detection algorithms is subjected to low signal-to-noise ratio [6]. Specifically, when the signal-to-noise ratio falls below 5 dB, the detection of R-peaks is considered unreliable [7]. Moreover, wearable devices are intended to be used during regular daily activities (walking, working, cooking etc.), which makes the Hearables signals collected by them, prone to increased presence of artifacts due to the subject movements. Even when poor quality or corrupted data are excluded from analysis, the accuracy of R-peak detectors applied for ambulatory data is worse than for data acquired in clinics [8]. Overall, the performance of R-peaks detectors decreases when they are applied to ambulatory data, compared to the case when the same detectors are applied to data acquired by physicians [9]. Georgiou *et al.* [10] emphasise that so far wearable devices can only be used as a surrogate for heart rate variability at resting or mild exercise conditions, as their accuracy fades out with increasing exercise load.

The solution for these problems may be the use of specially designed R-peak detectors such as a Deep matched filter (DMF), introduced by Davies *et al.* [11]. The DMF is a deep neural network model designed to search for matches with an ECG template pattern in the input signal. The DMF operates as a neural network based Matched Filter, trained to analyze in-ear data. The DMF showed higher median R-peak sensitivity (94.9%) and PPV (91.2%) than the conventional matched filter (62.3% and 67.0% respectively) and the Matched Filter Hilbert Transform (79.5% and 83.4%) in leave one-subject-out-cross validation performed on actual in-ear ECG signals acquired on 36 subjects,with an age range of 18-75 [11].

In this study, we used the DMF in a mobile tablet device for online R-peaks detection in in-ear ECG. We implemented the DMF on the Android mobile application, designed for acquired data from the BioBoard, a multisensors hearable platform developed in our lab. We performed computation time estimation of the implementation and analysis of DMF vulnerability to different types and levels of artefacts that disturb the ECG signal.

2. Deep Matched Filter

The DMF represents a combination of the encoder-decoder model and R-peak detector. The DMF was built in

two stages: first the encoder-decoder structure conssists of 1D convolutional layers, which was trained to transfer in-ear signal into ECG. In the second step, the R-peak detector on top of an encoder stage was trained. R-peak classifier stage includes a single-layer 1D convolution, followed by a Sigmoid activation function, flattening, and a linear output layer (Fig. 1). The DMF was previously trained using a separate dataset [11], we did not modify model weights.

The DMF is a PyTorch model created in Python, and must be prepared to work in a mobile application with PyTorch Lite interference. To do that a model must be serialized, optimized for mobile devices, and saved as PyTorch Lite model [12]. The procedure is described on the PyTorch website: https://pytorch.org/mobile/android/ (accessed 07.11.2023). In the mobile application we employed the detector part of DMF, which combines the encoder stage of the Deep Matched filter structure and R peaks classification layer (Fig. 1). The DMF PyTorch Lite model was stored inside the mobile application.

The DMF model execution pipeline can be divided into three stages: pre-processing, model running and post-processing (Fig. 2). In the pre-processing stage, signal filtering and data buffering are performed. We used IIR filter library (named iirj), available in the GitHub repository: https://github.com/berndporr/iirj.git (accessed: 07/11/2023) to implement bandpass and highpass filters. The library provides high-level API for filter design and filtering. We used three Butterworth filters: bandpass filter with cut-off frequencies 1 and 45Hz, bandpass filter from 1 to 5Hz and highpass filter with cut-off frequency of 1Hz. All filters have order = 4.

In the model-running stage, the input supplied to the DMF consists of a tensor with a size of (1, 3, 500) samples, with 500 samples originating from each filter. The DMF is trained to process signals at a sampling frequency of 250Hz, allowing it to analyze two seconds of recording in a single run. Nevertheless, the BioBoard records ECG signals at a sampling frequency of 500Hz. Resampling procedure was used to align the sampling frequency of ECG signal from our Bioboard (500Hz) and the DMF (250Hz). Subsequently, post-filtering, we retained only every other sample in circular buffers for each filter.

Davies *et al.* [11] introduced a mechanism of overlapping similar to Welch's method for spectrum estimation. They split a recording into overlapping segments of 500 samples, with a segment shift of 50 samples: the first segment contained samples from 1 to 500, the next segment from 51 to 550 and so on to the end of recording. Multiple results from overlapping runs for the same sample, were averaged and became DMF output. The overlapping mechanism reduced impact of sample position on the results, otherwise, the QRS complex migh QRS complex may be

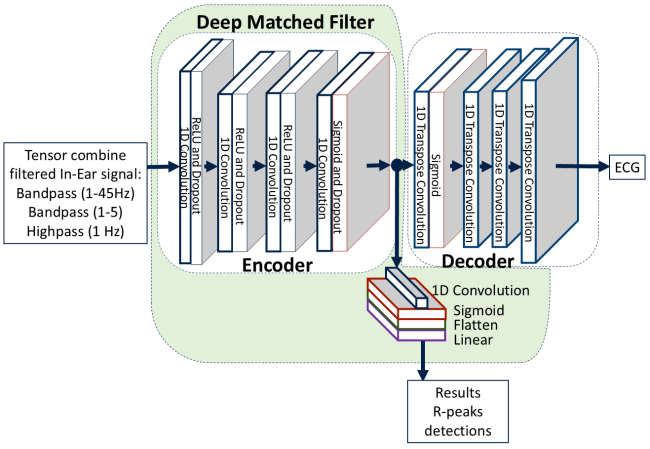

Figure 1. Deep Matched Filter structure; model includes convolution layers of the encoder decoder structure and the R-peak detector with convolution and fully connected layers.

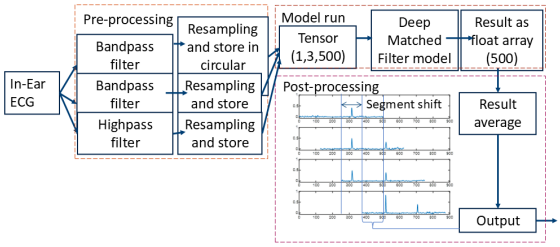

Figure 2. Stages of DMF model pipeline. In the pre-processing stage signal filtering and data buffering are performed; when the required number of samples are provided, defined by the segment shift parameter, the model run is performed. In post-processing Multiple results from overlay runs for the same sample are averaged and became DMF output.

split between two consecutive runs, and do not match the template. In the implementation, we made segment shift an adjustable parameter, set by a user in the application.

To run the PyTorch model in the Android application we used the PyTorch Android Lite library in version 1.13.1. The PyTorch Lite provides interference to load and run the prepared model. During the model run, input tensor was built from normalized filtered signals, then the forward run of the model was executed, and results were stored inside Float array. The results of the model run output were averaged in the post-processing stage (Fig. 2).

3. Method

We evaluated the computational time of the model run during 60-second recordings of ECG signal. We measured the computation time of different stages of the algorithms:

pre-processing, model run and post-processing (Fig. 2). To measure the computation time of DMF on Android we used *System.nanoTime()* function.

In this study, we used the Samsung Galaxy Tab S8 with Qualcomm SM8450 Snapdragon 8 Gen 1 chipset, Adreno 730 GPU, with Android 13. The chipset contains an AI engine for the fast processing of AI tasks.

Next, we assessed impact of common artefacts in ECG (baseline wander, muscle artefacts, and electrode motion artefacts) on the DMF performance. We evaluated DMF on artificial data, with a controlled level of signal-to-noise ratio (SNR). To generate data, we followed a procedure of creation of noisy recording described in mit-bih-noise-stress-test-database-1.0.0 [13]. We utilized recordings labelled 118 and 119 from the MIT-BIH Arrhythmia Database [14] and we used *nst* program from WFDB Toolbox to generate recordings with calibrated amounts of noise (SNR=24,18,12,6,0 and -6 dB); we used recordings of noise from MIT BIH noise stress test database (original database contains recordings only disturbed with electrode motion artifacts but now with baseline wander and muscle artifacts). The *nst* program adds noise beginning after the first 5 minutes of each record, during two-minute segments alternating with two-minute clean segments. In our analysis of detector performance, we used only segments of a record with added noise.

In our study, we define successful R-peak detection as peaks in the DMF output that exceed an amplitude of 0.15. A detection is classified as a true positive if it occurs within a 100 ms window (either before or after) the QRS annotation in the original recording. Additionally, to investigate whether decreasing the segment size can enhance the DMF's performance, we conducted repeated DMF evaluations, each time using a different segment shift parameter.

4. Results

The average time of the whole analysis on the device was 7.46 ms. The DMF model run took 74% of time (5.515 ms); post-processing and aggregate results took almost 26% of run time (1.938ms); pre-processing, flirtations, and storing samples in buffers took only 0.007 ms.

The segment shift parameter defines how many the model runs results will be averaged to obtain the final output. Fig. 3 shows sensitivity and positive predictive value of DMF in the function of segment shift. The overall performance of the model slightly increased when segment shifts decreased. The sensitivity and PPV of DMF somewhat decreased when baseline wander artefacts were applied. In that case when SNR decreased from 24 to -6 sensitivity decreased from 0.87 to 0.81 and PPV from 1.00 to 0.96. Muscle artifacts and electrode motion artefacts affected the performance of the model. In the case of electrode motion artefacts sensitivity decreased from 0.87 to 0.19 and PPV from 1.00 to 0.44 respectively. For muscle artifacts, sensitivity decreased from 0.87 to 0.30 and PPV from 1.00 to 0.64 (segment shift = 50).

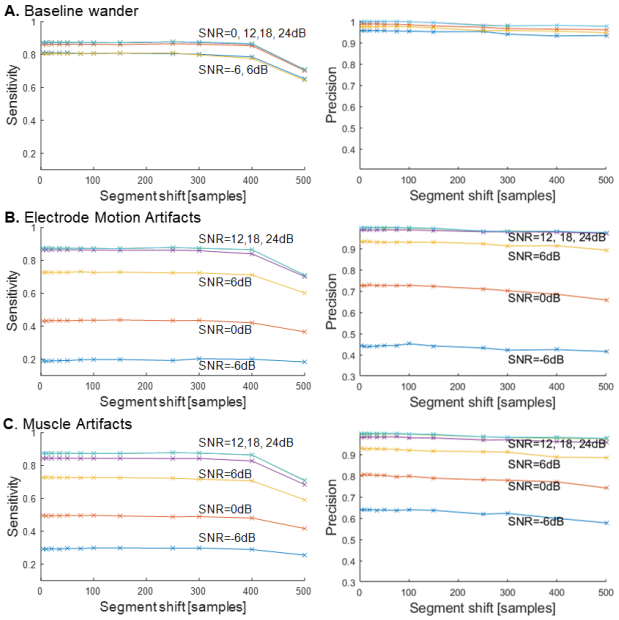

Figure 3. Sensitivity (on the left) and positive predictive value (right side) of DMF as a function of segment swift parameter for different signal to noise levels and artifacts type: A. Baseline wander, B. Electrode motion artifacts and C. Muscle artifacts.

5. Discussion

Implementing a neural network model developed in PyTorch into a mobile application via PyTorch Lite was relatively straightforward. This framework offers a high-level API for executing neural network models. Programmers are required to write specific procedures for preparing the input data and handling the output. Additionally, the Android system includes a Neural Networks API that supports hardware-accelerated inference, compatible with PyTorch Lite. According to Google, this neural network runtime on Android devices is capable of efficiently distributing computational tasks across the device's processors. This includes specialized neural network hardware, graphics processing units, and digital signal processors.

The Samsung Galaxy Tab S8 provides enough computation power to run the DMF in real-time. Without any changes, the DMF can be run 134 times per second, this is enough to perform model run overlapping, with minimal value of segment shift parameter set as 2.

Reduced segment shift parameter results in an improvement of the DMF performance (Fig. 3), but it also in-

creased computational workload. Reducing the parameter to below 50 samples provided a marginal increase in DMF's sensitivity and PPV but it also slightly increased average computation time. On the other hand, increasing the segment shift to above 300 samples resulted in a drop in model performance.

The DMF sensitivity is lower than PPV for given SNR (Fig. 3), which indicates that the model rather misses R-peak deductions than provides an extra one. In this study, we used simple constant threshold peak detection. The model performance may be improved by the use of an adaptive threshold, and look-back mechanism, similar to that used in the Pan-Tompkins algorithm.

The DMF dealt well with baseline wander artefacts. Reduced SNR in that case slightly affected the performance of the detector. Regrettably, the DMF's performance dropped for electrode motion artefacts and muscle artefacts in the case of SNR 6 dB and below. Since the DMF is based on a neural network model and requires training, the performance of the model may be improved by retraining the model with artificial noise recording.

There are some possible improvements of the DMF. Further optimization of the DMF may reduce the size, memory footprint and computation time of the model. Standard strategies for a neural network model optimization, reduction of model complexity without losing of its accuracy, include removing small weight connections (pruning) and quantization, reduction of the number of bits that code weights [15]. Another way of improving model performance and reduction of its size can be achieved using loss functions dedicate to the task. Brophy *et al.* [16] trained ECG artefact removal network, based on convolutional layers, with loss function defined as a sum of mean square errors for the whole signal and means square error calculated only for QRS parts of the signal. The researchers founded that the output SNR improved by 0.5 dB when compared to the standard mean square error function. Additionally, their custom function enabled a reduction of the neural network's weights by nearly 1,500 times.

References

[1] Nakamura T, Alqurashi YD, Morrell MJ, Mandic DP. Hearables: automatic overnight sleep monitoring with standardized in-ear EEG sensor. IEEE Transactions on Biomedical Engineering 2019;67(1):203–212.

[2] Hammour G, Yarici M, von Rosenberg W, Mandic DP. Hearables: Feasibility and validation of in-ear electrocardiogram. In Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2019; 5777–5780.

[3] Passler S, Müller N, Senner V. In-ear pulse rate measurement: a valid alternative to heart rate derived from electrocardiography? Sensors 2019;19(17):3641.

[4] von Rosenberg W, Chanwimalueang T, Goverdovsky V, Peters NS, Papavassiliou C, Mandic DP. Hearables: Feasibility of recording cardiac rhythms from head and in-ear locations. Royal Society Open Science 2017;4(11):171214.

[5] Yarici M, Von Rosenberg W, Hammour G, Davies H, Amadori P, Ling N, Demiris Y, Mandic DP. Hearables: feasibility of recording cardiac rhythms from single in-ear locations. Royal Society Open Science 2024;11(1):221620.

[6] Fariha MAZ, Ikeura R, Hayakawa S, Tsutsumi S. Analysis of pan-tompkins algorithm performance with noisy ecg signals. Journal of Physics Conference Series jun 2020; 1532(1):012022.

[7] Smital L, Haider CR, Vitek M, Leinveber P, Jurak P, Nemcova A, Smisek R, Marsanova L, Provaznik I, Felton CL, et al. Real-time quality assessment of long-term ECG signals recorded by wearables in free-living conditions. IEEE Transactions on Biomedical Engineering 2020; 67(10):2721–2734.

[8] Liu F, Liu C, Jiang X, Zhang Z, Zhang Y, Li J, Wei S. Performance analysis of ten common QRS detectors on different ECG application cases. Journal of Healthcare Engineering 2018;(1):9050812.

[9] Khamis H, Weiss R, Xie Y, Chang CW, Lovell NH, Redmond SJ. QRS detection algorithm for telehealth electrocardiogram recordings. IEEE Transactions on Biomedical Engineering 2016;63(7):1377–1388.

[10] Georgiou K, Larentzakis AV, Khamis NN, Alsuhaibani GI, Alaska YA, Giallafos EJ. Can wearable devices accurately measure heart rate variability? A systematic review. Folia Medica 2018;60(1):7–20.

[11] Davies HJ, Hammour G, Zylinski M, Nassibi A, Stanković L, Mandic DP. The deep-match framework: R-peak detection in ear-ecg. IEEE Transactions on Biomedical Engineering 2024;.

[12] Żyliński M, Nassibi A, Rakhmatulin I, Malik A, Papavassiliou C, Mandic DP. Deployment of artificial intelligence models on edge devices: A tutorial brief. IEEE Transactions on Circuits and Systems II Express Briefs 2023;1–1.

[13] Moody G, Muldrow W, Mark R. The MIT-BIH noise stress test database. Computers in Cardiology 1984;381–384.

[14] Moody GB, Mark RG. The impact of the MIT-BIH arrhythmia database. IEEE Engineering in Medicine and Biology Magazine 2001;20(3):45–50.

[15] Zhao T, Xie Y, Wang Y, Cheng J, Guo X, Hu B, Chen Y. A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities. volume 110. IEEE, 2022; 334–354.

[16] Brophy E, Hennelly B, De Vos M, Boylan G, Ward T. Improved electrode motion artefact denoising in ECG using convolutional neural networks and a custom loss function. IEEE Access 2022;10:54891–54898.

Address for correspondence:

Marek Żyliński
Department of Electrical and Electronic Engineering,
Imperial College London, SW7 2AZ London, U.K.
m.zylinski@imperial.ac.uk